UTILITY PATENT APPLICATION
UNDER 37 CFR 1.53(b)

Box PATENT APPLICATION
Assistant Commissioner for Patents
Washington, DC 20231

Case Docket No. 43889-964

Sir:

Transmitted herewith for filing is the patent application of:

INVENTOR: Makoto FUJIWARA
FOR: BUS STRUCTURE, DATA BASE AND METHOD OF DESIGNING INTERFACE

Enclosed are:

- ☒ 27 pages of Japanese language specification, claims, abstract.
- ☐ Declaration and Power of Attorney.
- ☒ Priority Claimed.
- ☐ Certified copy of _____
- ☒ 13 sheets of formal drawing.
- ☐ An assignment of the invention to _____ and the assignment recordation fee.
- ☐ An associate power of attorney.
- ☒ Information Disclosure Statement, Form PTO-1449 and reference.
- ☒ Return Receipt Postcard
- ☐

Respectfully submitted,

MCDERMOTT, WILL & EMERY

Michael E. Fogarty
Registration No. 36,139

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 MEF:dtb
Date: July 20, 2000
Facsimile: (202) 756-8087

バス構造，データベース及びインターフェースの設計方法

発明の背景

本発明は、システムＬＳＩなどの半導体集積回路装置中のバスの構造，インターフェースの設計方法及びインターフェースの設計などに用いるためのデータベースに関する。

従来より、半導体回路のＣＰＵとこれによって制御される回路との間を接続するインターフェースと呼ばれる部分は、ＣＰＵや各回路間の通信を行なう上で重要な部分である。このインターフェースの中心となる部分は“バス”と呼ばれる信号線であり、データを送る際にはこのバスへのアクセス権をいかに獲得するか、などのデータの入出力を制御するためのシステムが重要である。すなわち、バス構造も含めたインターフェースは最終的なデバイスの性能を大きく左右する要素である。

ここで、図１に示すように、従来のバスの構造としては、ノイマン系プロセッサに用いられるノイマン系バス構造と、ハーバード系プロセッサに用いられるハーバード系バス構造とが知られている。ノイマン系バス構造は、アドレスとデータとのみを区別して、アドレスとデータとを１つの線で表現したり、アドレスとデータとを２つの線で表現する構造のものである。ノイマン系バス構造として、アドレス及びデータを共通のバスを介して流通させるマルチプレクサ型と、アドレスとデータとを別系統のバスつまりアドレスバスとデータバスとを介して個別に流通させるデマルチプレクサ型とが知られている。

また、ハーバード型プロセッサとは、データに対して制御データと実際の転送ファイルのデータとを内容によって分離するための構造である。従来開発されているハーバード系バス構造としては、アドレスバスをさらにＩＯアドレスバスとメモリアドレスバスとに分け、データバスを制御データバスと転送データバスとに分けたもの（以下、これを情報分離型という）が知られている。

マルチプレクサ型バス構造は、アドレス，データの制御と転送とを逐次処理していくために用いられるバス構造であり、これを用いたときの処理能力は比較的

低いがバスの占有面積（バンド幅）は小さい。デマルチプレクサ型バス構造は、アドレスの制御及び転送と、データの制御及び転送とを並列的に処理していくために用いられるバス構造であり、これを用いたときには並列処理が可能なので処理スピードは高くなる。

また、従来のハーバード系バス構造である情報分離型バス構造は、アドレスとデータの双方について、それぞれ制御と転送とを並列的に処理していくために用いられるバス構造であり、これを用いたときの処理能力はさらに高くなる。

ところが、従来のシステムＬＳＩなどの大規模のデバイスを構築する際に、インターフェースの構造をいかに構築するかという点については、未だ適切な手法が確立されていないのが現状である。すなわち、バス構造だけについてみても、各種のバス構造の利点と欠点とがあり、これらを各回路の動作との関連で統一的に評価するための手法は未だ確立されていない。

また、半導体集積回路装置の規模が大きくなり、例えばシステムＬＳＩと呼ばれる複数の半導体回路を組み合わせたデバイスを設計するようになると、上記従来のようなバス構造だけでは、インターフェースを設計する際に、バス構造をフレキシブルに利用できないという不具合もある。

発明の概要

本発明の目的は、システムＬＳＩなどの大規模の半導体デバイスのインターフェースを構築するための設計方法と、その設計に用いるためのバス構造やデータベースを提供することにある。

本発明のバス構造は、半導体集積回路中の制御回路と複数の被制御回路とを接続するためのバス構造であって、上り側バスと下り側バスとに分離されたアドレスバスと、上り側バスと下り側バスとに分離されたデータバスとを備えている。

これにより、ある被制御回路にデータを送りながら他の被制御回路に別のデータを送るような同時命令処理に対する制限が緩和され、バス構造を用いたデバイスのデータ処理能力が向上することになる。

上記バス構造において、上記データバスを、上記被制御回路ごとに分離し、かつ、被制御回路ごとに上り側バスと下り側バスとに分離しておくことにより、さらに同時命令処理に対する制限が緩和される。

本発明のデータベースは、半導体集積回路の設計のために利用されるデータベースであって、制御機能部と複数のアプリケーションとの間を接続するバス構造の種類に関する記述が含まれたテーブルを備えている。

これにより、バス構造によって異なる同時命令処理に対する制限や電力，面積などを総合的に考慮したインターフェースの設計に供しうるデータベースが得られる。

上記データベースにおいて、上記テーブルに、各アプリケーションの動作モデルによって実現される性能を評価するための性能指数を記述した性能テーブルを備えることにより、システム全体の性能の評価を含むインターフェースの構築が可能になる。

上記データベースにおいて、上記性能テーブルは、上記性能指数として、処理能力，バス幅，命令量及びメモリ量のうち少なくともいずれか１つのパラメータを含むことにより、バス構造の種類によって異なるこれらのパラメータを考慮したインターフェースの構築が可能になる。

上記データベースにおいて、上記性能テーブルは、バス構造に関する記述として、上り側バスと下り側バスとに分離されたアドレスバスと、上り側バスと下り側バスとに分離されたデータバスとからなる分離型バス構造に関する記述を含んでいることにより、新たなバス構造を利用したインターフェースの構築が可能になる。

本発明の第１のインターフェースの設計方法は、複数のアプリケーションごとにその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって、上記アプリケーションの動作モデルとして上記複数のライブラリを順次用い、上記各ライブラリについて、上記制御機能部により各アプリケーションを無制限に動作させる動作シミュレーションを行なって、バストランザクションの衝突回数を解析する方法である。

この方法により、アプリケーションを動作させたときにバスの混雑度がライブラリの選択によってどう変わるかを考慮して、インターフェースの設計を行なうことが可能になる。

上記第１のインターフェースの設計方法において、各アプリケーション同士の間に上記トランザクションの衝突回数に応じた段数のＦＩＦＯを生成し、ＦＩＦＯを仮挿入した状態で上記解析を行なうことにより、トランザクションの衝突を回避したときの性能を考慮してインターフェースの設計を行なうことが可能になる。

本発明の第２のインターフェースの設計方法は、複数のアプリケーションごとにその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって、上記複数のアプリケーションの動作モデルとして上記複数のライブラリを順次用い、上記各ライブラリについて、上記制御機能部により各アプリケーションを無制限に動作させる動作シミュレーションを行なって、同時命令処理の回数を解析する方法である。

この方法によって、アプリケーションを動作させたときのシステムの処理能力がライブラリの選択によってどう変わるかを考慮して、インターフェースの設計を行なうことが可能になる。

上記第２のインターフェースの設計方法において、上記同時命令数に応じて、クロスバスの構造を決定することにより、制御機能部の負荷率の低減や電流値の分散を図ったときの性能を考慮してインターフェースの設計が可能になる。

上記第２のインターフェースの設計方法において、さらに、上記同時命令数が所定値以上であるバスに配置するための転送動作の制御機能部を生成し、転送動作の制御機能部を配置した状態で上記解析を行なうことにより、並列的に転送動作を可能な状態にしたときの性能を考慮したインターフェースの設計が可能になる。

本発明の第３のインターフェースの設計方法は、複数のアプリケーション及び複数のバス構造の各々に対してその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって、上記半導体集積回路の価値を最終的に評価するための複数の主パラメータを設定し、上記主パラメータに対して影響を与える複数の副パラメータを設定するステップ（ａ）と

、上記各主パラメータごとに、各ライブラリの副パラメータによって主パラメータを評価して、主パラメータの目標値を満たすライブラリ群を選び出すステップ（ｂ）と、上記選び出されたライブラリ群ごとに定まる複数の主パラメータを評価して、最適なライブラリ群を選ぶことにより、インターフェースを決定するステップ（ｃ）とを含んでいる。

この方法により、一度にすべてのパラメータから性能を評価するのに比べ、より統一的な手法で最適なライブラリ群を選び出し、最終的に最適なインターフェースを決定することができる。

上記第３のインターフェースの設計方法において、上記ステップ（ａ）の前に、上記複数のアプリケーションの動作モデルとして上記複数のライブラリを順次用いて、上記各ライブラリについての動作シミュレーションを行なうことにより、より正確に最適なインターフェースを決定することができる。この方法の具体的な手法としては、以下のような手法がある。

例えば、上記ステップ（ａ）では、３つの主パラメータを設定し、各主パラメータごとに３つの副パラメータを設定して、上記ステップ（ｂ）では、３つの副パラメータを座標軸とする３次元座標系を構成して、各副パラメータの値によって定まる３角形の面積が目標値以下のライブラリ群を選び出し、上記ステップ（ｃ）では、３つの主パラメータを座標軸とする３次元座標系を構成して、選び出されたライブラリ群から求まる主パラメータの値によって定まる３角形の面積が最小のライブラリ群からインターフェースを決定することができる。

また、上記ステップ（ａ）の後ステップ（ｂ）の前に、上記複数の副パラメータのうち特定の副パラメータに着目して、特定の副パラメータが目標値を満たしているライブラリ群を選び出し、上記ステップ（ｂ）では、上記複数の主パラメータのうち特定の主パラメータを除く主パラメータが目標値を満たしているライブラリ群を選び出し、上記ステップ（ｃ）では、上記特定の主パラメータが最小となるライブラリ群を最適なライブラリ群として選び出すこともできる。

さらに、上記ステップ（ａ）では、主パラメータに対する複数の副パラメータの影響係数を設定し、上記ステップ（ｂ）では、影響係数と副パラメータの値とに基づいて各主パラメータの目標値を満たすライブラリ群を選び出し、上記ステ

ップ（ｂ）では、選び出されたライブラリ群から求まる複数の主パラメータに重み付けを行なってから、ライブラリ群を決定するための演算を行なうこともできる。

本発明の第４のインターフェースの設計方法は、複数のアプリケーション及び複数のバス構造の各々に対してその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって、上記複数のアプリケーションの動作モデルとして上記複数のライブラリを順次選択するステップ（ａ）と、上記制御機能部により上記各アプリケーションを動作させて、各ライブラリを用いたときの制御機能部，インターフェース及びアプリケーションの性能を解析するステップ（ｂ）と、上記ステップ（ａ）とステップ（ｂ）とを繰り返し行なって、上記解析結果に基づいて最適なライブラリ群を選ぶことにより、インターフェースを決定するステップ（ｃ）と、上記決定されたパラメータに基づいて最適なインターフェースを合成するステップ（ｄ）とを含んでいる。

この方法により、各アプリケーションを動作させたときのシステム全体の性能を評価した上で、最適なインターフェースを合成することが可能になり、インターフェース設計の基本的な手法を確立することができる。

上記第４のインターフェースの設計方法において、上記ステップ（ｂ）では、上記各ライブラリについて、上記制御機能部により各アプリケーションを無制限に動作させたときのバストランザクションの衝突回数を解析し、上記ステップ（ｄ）では、各アプリケーション同士の間に上記トランザクションの衝突回数に応じた段数のＦＩＦＯを挿入することが好ましい。

上記第４のインターフェースの設計方法において、上記ステップ（ｂ）では、上記各ライブラリについて、上記制御機能部により各アプリケーションを無制限に動作させたときの同時命令処理の回数を解析し、上記ステップ（ｄ）では、上記同時命令数が所定値以上であるバスにはクロスバスを配置することをが好ましい。

図面の簡単な説明

図１は、従来のノイマン系バス構造，従来のハーバード系バス構造及び本発明

の第１の実施形態のハーバード系バス構造の構造上の相違を概略的に示すブロック図である。

図２（ａ）～（ｃ）は、従来のバス構造，第１の本実施形態の方向分離型バス構造（３次局がある場合及びない場合）の例を順に示す図である。

図３（ａ）～（ｄ）は、順に、ノイマン系のマルチプレクサ型，デマルチプレクサ型，ハーバード系の情報分離型及び第１の実施形態の方向分離型バス構造におけるアドレス及びデータの処理を時間軸で示す図である。

図４は、第１の実施形態における対象リソースＡがメモリで、対象リソースＢ，ＣがＩＯである場合のリソース分離型バス構造を示すブロック図である。

図５（ａ），（ｂ）は、第２の実施形態における設計用のデータベースに記憶されている単体アプリケーションＡ，Ｂでのライブラリ及び性能テーブルの例を示す図である。

図６は、第２の実施形態における複数のアプリケーションを用いて動作シミュレーションを行なう方法の例を示すブロック図である。

図７は、第２の実施形態における性能解析のうちのトランザクション解析の表示方法を示す図である。

図８は、第２の実施形態における性能解析のうちの命令処理解析の表示方法を示す図である。

図９（ａ），（ｂ）は、第２の実施形態において同時命令の多い箇所にクロスバスを設けた場合のバス構造の一部を示す図である。

図１０（ａ）～（ｄ）は、各パラメータが目標値を満たしているライブラリの中で交差面積が最小のものを選択する方法の手順を示す図である。

図１１（ａ）～（ｄ）は、性能指数，平均電力指数（又は最大電力指数）及び面積指数についてそれぞれ重み付け行なってから加算し、その加算値である最適指数が最小のものを選択する方法である。

図１２は、第２の実施形態のインターフェースの設計方法を用いて合成される最適ＩＦの構造を示すブロック図である。

図１３は、第２の実施形態における最適ＩＦの合成を含むシステム設計の手順を示すフローチャート図である。

好適実施形態の説明

（第１の実施形態）

図１は、従来のノイマン系バス構造と、従来のハーバード系バス構造と、本実施形態のハーバード系バス構造との構造上の相違を概略的に示すブロック図である。

本実施形態におけるハーバード系バス構造は、“方向分離型バス構造”というべきものであって、情報分離型バス構造における転送データをさらに分離して処理するものといえる。この方向分離型バス構造は、プロセッサ（制御回路）を中心として対象リソース（被制御回路）を配置したときの転送データが送られる方向を、“上り”と“下り”とに分けるものであるが、さらに、転送データの被制御回路をメモリ用データとＩＯ用データとに分ける場合もある。ここで、制御データとは、例えば認識，応答などのコントロールの情報を持ったものである。転送データとは、例えば画像ファイルのデータなどのまとまったデータをいう。

図２（ａ）は、従来のバス構造を示す図である。同図に示すように、従来のバス構造では、ＣＰＵと対象リソース（ここでは１次局ＩＯ）との間には１つのバスしか設けられていなかった。

それに対し、本実施形態の方向分離型バス構造においては、図２（ｂ），（ｃ）に示すように、プロセッサ（ＣＰＵ）と一次局との間の通信をも、上り側バスと下り側バスとに分けて行なうように構成されている。ここで、図２（ｂ）は、各々１次局を挟んでＣＰＵと通信を行なう２次局と３次局とが存在している場合の方向分離型バス構造を示す図であり、図２（ｃ）は、図２（ｂ）における３次局が存在しない場合の方向分離型バス構造を示す図である。

図３（ａ）～（ｄ）は、順に、ノイマン系のマルチプレクサ型バス構造，ノイマン系のデマルチプレクサ型バス構造，ハーバード系の情報分離型バス構造及び方向分離型バス構造（方向型）におけるアドレス及びデータの処理を時間軸で示す図である。

図３（ａ）に示すように、マルチプレクサ型バス構造は、アドレスとデータとをいわば１つの線上でシリアルに処理していくためのバス構造である。例えば対象リソースＡに対して何らかのコマンドを生成する場合に、対象リソースＡの制

御アドレスを指定し、対象リソースＡの制御データを送り、対象リソースＡに転送するためのアドレスを送り、次にＡへの転送データを送る。“Ｂ”に対するコマンドを生成する際にも、同様の手順による処理を逐次（シリアルに）行なうのである。

図３（ｂ）に示すように、デマルチプレクサ型バス構造は、アドレスとデータとを並列的に処理していくためのバス構造である。例えば対象リソースＡに対して何らかのコマンドを生成する場合に、対象リソースＡの制御アドレスを指定しながらＡの制御データを送り、Ａに転送するためのアドレスを送っている間に対象リソースＡへの転送データを送る。対象リソースＢに対するコマンドを生成する際にも、同様の手順による処理を並列的に行なうのである。

図３（ｃ）に示すように、情報分離型バス構造は、制御アドレス，転送アドレス，制御データ及び転送データをそれぞれ並列的に処理していくためのバス構造である。例えば対象リソースＡに対して何らかのコマンドを生成する場合に、対象リソースＡの制御アドレスの指定と、対象リソースＡへの制御データの送信と、対象リソースＡに転送するためのアドレスの送信と、対象リソースＡへの転送データの送信とを並列的に処理する。対象リソースＢに対するコマンドを生成する際にも、同様の手順による処理を並列的に行なう。

図３（ｄ）に示すように、情報分離型バス構造は、制御アドレス，転送アドレス，制御データ及び転送データをそれぞれ並列的に処理していくに加えて、アドレス及びデータの転送を、上りと下りとに分けて並列的に行なうためのバス構造である。例えば対象リソースＡ，Ｂ，Ｃに対して何らかのコマンドを生成する場合に、対象リソースＡ，Ｂ，Ｃの制御アドレスを指定するとともに上りアドレスと下りアドレスとも指定しておく。そして、上りアドレスを指定しているときには上り側バスを介して上り側の転送アドレス及び転送データ（例えば対象リソースＡ，Ｃについてのもの）の送信を、下りアドレスを指定しているときには下り側バスを利用して下り側の転送アドレス及び転送データ（例えば対象リソースＢ，Ａについてのもの）を送信することができる。言い換えると、制御アドレスの指定や制御データの送信とは無関係に、転送アドレスの送信と転送データの送信とを行なうことができる。

例えば、図２（ｂ）に示すバス構造を有する場合、１次局ＩＯ１が対象リソースＡで、１次局ＩＯ２が対象リソースＢで、２次局が対象リソースＣであるとする。このとき、対象リソースＡ（１次局ＩＯ１）のアドレスを指定している際にも、対象リソースＢ（１次局ＩＯ２）には対象リソースＡ（１次局ＩＯ１）のアドレスの指定が入力されないので、転送アドレスＡを指定して転送データを対象リソースＡに転送している間に、転送アドレスＢを指定して転送データを対象リソースＢに送ることが可能になる。そして、データをインプット（上り）すると、すぐにデータをアウトプット（下り）することが可能になる。

図４は、対象リソースＡがメモリで、対象リソースＢ，ＣがＩＯである場合のリソース分離型バス構造を示すブロック図である。対象リソースＣ（ＩＯ２）からのデータの入力を受け、対象リソースＡ（メモリ）にこれを記憶させながら、対象リソースＢ（ＩＯ１）にデータを転送するという動作が同時に行えることになる。

その結果、従来の情報分離型バス構造を用いた場合には、データ転送に要する時間は、図３（ｄ）に示すように、データ転送Ａ，データ転送Ｂ，データ転送Ｃ及びデータ転送Ａに要する時間をシリアルに加算したものになる（転送Ａ，Ｂ，Ｃ，Ａアドレスの時間も同じ）が、本実施形態の方向分離型バス構造を用いた場合には、図３（ｄ）に示すように短縮されることになる。つまり、上り側バスと下り側バスとに分けることによって、対象リソースＡ，Ｃに対する上り側の転送アドレス及び転送データの送信と、対象リソースＢ，Ａに対する下り側の転送アドレス及び転送データの送信とを同時に行なうことが可能になるからである。

（第２の実施形態）

次に、バス構造をも含めたＩＦの設計方法に関する第２の実施形態について説明する。ここでは、最適なＩＦを設計する手順について説明する。

－準備するデーター

ここでは、まず性能解析を行なうために必要なデータ（ライブラリモデル）について説明する。単体アプリケーションごとに、動作モデルを作成する必要があるが、その動作モデルの１つの作成方法として、基本的にソフトウェアで作成し、あるものはここまでをハードウェアで作成し、他のものは別の部分までをハー

ドウェアで行なうように決定する。なお、単体アプリケーションとは、１つのアプリケーションの入出力関係を定義したもので、範囲の大きさは定義していない。例えば、単体アプリケーションの例として、ＩｒＤＡを用いたプリントデータ転送の動作や、ＵＳＢを用いたマウスの位置データの転送や、ＪＰＥＧを用いた静止画像データの圧縮，解凍動作などがある。ここで、“ＩｒＤＡ”と呼ばれる赤外線通信を用いてデータの転送についてのアプリケーションを使用するときには、コンピュータ側で例えばプリントするデータを事前に加工しなければならない。例えば、静止画を圧縮するためにＪＰＥＧで圧縮する加工を施して、そのデータを赤外線として表現するためのフォーマット変換を行なってから、データを送るという赤外線通信を行なうことになる。その場合に、ここでいう単体アプリケーションとは、赤外線通信（ＩｒＤＡ）を行なう前の処理は除いて、ＩｒＤＡと呼ばれているアプリケーションの入力と出力とを規定している部分だけを意味している。

ここで、動作モデルをハードウェアで作成するか、ソフトウェアで作成するかによって処理時間が異なってくる。本実施形態では、１つのプロトコルの表現において各層がある程度明確に分かれている部分で、動作モデルをハードウェアで作成するかソフトウェアで作成するかを分ける。

図５（ａ），（ｂ）は、設計用のデータベースに記憶されている単体アプリケーションＡ，Ｂでのライブラリ及び性能テーブルの例を示す図である。例えば、アプリケーションＡ，Ｂに対して、仕様レベルのライブラリＡ，Ｂがあるとすると、動作レベルにおいては、同図に示すように、アプリケーションＡ，Ｂごとに、ＣＰＵ（バス構造）として、ノイマン型ＣＰＵ（バス構造），従来のハーバード型ＣＰＵ（バス構造），方向分離型ＣＰＵ（バス構造）を採用したときの動作を記述した動作モデルとなるライブラリが登録されている。そして、各ライブラリの性能テーブルにおいては、各ライブラリの性能指数が、処理能力（Ｐ），バス幅（Ｂ），命令量（Ｍ），メモリ量（Ｅ）という変数（パラメータ）の関数として表されている。

図５（ａ），（ｂ）において、ライブラリの動作を記述した部分のうちハッチング部分がソフトウェアによって実現する部分を、白地の部分がハードウェアに

よって実現する部分をそれぞれ示している。例えばアプリケーションＡにおいて、各機能のすべてをソフトウェアで実現すると性能指数が１００になるとすると、ハードウェアで４０％をソフトウェアで残りの６０％をそれぞれ実現させる場合には、例えば性能指数が５０になり、ハードウェアで６０％を実現した場合には、性能指数が１０になる。このように、ソフトウェアのうちの何％をハードウェアで置き換えると性能指数がいくらで済むかについての性能テーブルが準備されている。

ここで、動作モデルの機能のうち，“ＦＬＯＷ”とは、例えばａ，ｂという処理がある場合に、“処理ａを行なってから処理ｂを行なう”というような処理のフローを記述した層をいう。“ＭＡＮＧ”とは、例えば、別々のファイル転送アプリケーション同士が接続されているときに、その各々に必要なデータ交換のマルチプレクサなどのアプリケーション間の通信をマネジメントする方法を記述した層をいう。“ＬＩＮＫ”とは、例えば１つの情報データが、同期ビット，コントロールデータ，ＭＡＣデータ，終了などの一連のデータによって囲まれた範囲をデータとして認識させるというような，接続の手順を定めている層をいう。“ＰＨＹ”とは、実際のコーディングの仕方、例えば“１”を表現するときに、あるパルス幅においてすべて“１”になっているか、あるパルス幅の中央部で“１”になっているか、などを区別している層をいう。また、“ＣＡＬ”とは、例えば演算処理の場合には、演算処理（かけ算など）をハードウェアで行なうのか、ソフトウェアで行なうのかを示す層をいう。

また、ソフトウェアで行なう場合には高い性能が必要となるが、ハードウェアがある程度あることによって性能が低くても済むことが多いことに留意しておく必要がある。そして、図５（ａ），（ｂ）に示すような性能テーブルが、多数のアプリケーションについて準備されているが、図５（ａ），（ｂ）にはアプリケーションＡ，Ｂのみが例示されている。

－動作シミュレーション－

図６は、複数のアプリケーションを用いて動作シミュレーションを行なう方法の例を示すブロック図である。ここでは、アプリケーションＡには１００％ソフトウェア化したライブラリを、アプリケーションにＢは２０％だけハード化した

ライブラリ、アプリケーションＣには４０％だけハード化したライブラリを用いると仮定して、この３つのアプリケーションの転送処理の性能解析を行なうとする。

ここで、図６に示す接続関係において、インプット，アウトプットをカウントする部分（入出力カウント部）と、命令処理をカウントする部分（命令処理カウント部）とを、動作解析シミュレーション中に記述しておく。入出力カウント部とは、例えば、転送処理アイル（システム動作制御）に対してアプリケーションＡ，Ｂ，Ｃが接続されている場合に、何も処置を施さずに転送処理を行なったとすると、プロセッサと各機器との間における入出力が互いに衝突（バストランザクションの衝突）する回数をカウントする。つまり、バスの混雑度を測定するのである。また、命令処理がどのぐらいの数だけ発生しているかを、その衝突も含めてカウントしておく。

図７は、この性能解析のうちのトランザクション解析の表示方法を示す図である。同図に示すように、各アプリケーションＡ，Ｂ，Ｃについてノイマン型バス構造（デマルチプレクサ型）のライブラリを採用した場合と、従来のハーバード型バス構造（情報分離型）のライブラリを採用した場合と、方向分離型バス構造のライブラリ採用した場合とについて、入出力カウント部で測定されたアプリケーションＡ－Ｂ間、アプリケーションＢ－Ｃ間、アプリケーションＡ－Ｃ間のトランザクション密度を処理時間軸の方向にまとめる。図７において、ハッチングの密度が濃いほど衝突回数が多くなるものとする。ノイマン型バス構造を採用した場合は、並列処理を行なうとどうしても衝突が多くなることがわかる。それに対し、方向分離型バス構造を採用した場合は、並列処理が容易な構造であるので、衝突回数は比較的少なくなる。そして、この衝突回数の多い箇所にＦＩＦＯを挿入する。例えば、ノイマン型バス構造を採用する場合には、Ａ－Ｂ間にＦＩＦＯをｋ段（例えば１０段）挿入し、Ｂ－Ｃ間にＦＩＦＯを１段（例えば８段）挿入し、Ａ－Ｃ間にＦＩＦＯを１段挿入する。また、従来のハーバード型バス構造（情報分離型）を採用した場合には、Ａ－Ｂ間にＦＩＦＯを１段挿入し、Ｂ－Ｃ間にＦＩＦＯをｍ段（例えば６段）挿入し、Ａ－Ｃ間にＦＩＦＯをｍ段挿入する。また、方向分離型バス構造を採用した場合には、Ａ－Ｂ間にＦＩＦＯをｎ段（

例えば４段）挿入し、Ｂ－Ｃ間にＦＩＦＯをｎ段挿入し、Ａ－Ｃ間にもＦＩＦＯをｎ段挿入する。ＣＰＵが処理している間に、ＦＩＦＯで情報をためておく処理が必要となるが、バスの混雑度に応じてＦＩＦＯの段数を解析しておくのである。もちろん、バストランザクションの衝突回数が少ないほど、挿入すべきＦＩＦＯの段数は少なくなる。このようにして、それぞれのバス構造の構造つまりＣＰＵにつながるバスの構造に応じて、ＦＩＦＯを何段挿入する必要があるかが解析される。

次に、図８は、この性能解析のうちの命令処理解析の表示方法を示す図である。同図に示すように、アプリケーションＡ，Ｂ，Ｃについて、ノイマン型バス構造（デマルチプレクサ型）のライブラリを採用した場合と、従来のハーバード型バス構造（情報分離型）のライブラリを採用した場合と、方向分離型バス構造のライブラリを採用した場合とについて、命令処理カウント部で測定されたアプリケーションＡ－Ｂ間、アプリケーションＢ－Ｃ間、アプリケーションＡ－Ｃ間の同時命令密度を処理時間軸の方向にまとめる。図８において、ハッチングの密度が濃いほど同時命令回数が多くなり、システム全体の処理能力が高いことを示している。ノイマン型バス構造を採用した場合は、同時命令回数が少ないことがわかる。それに対し、方向分離型バス構造を採用した場合は、同時命令回数が比較的多くなる。同時命令回数が多い場合には、処理時間が短くなり応答スピードが速くなるが、反面、ＣＰＵの負荷率が増大し、瞬間的な電流値も増大するという不具合もある。

図９（ａ），（ｂ）は、この同時命令の多い箇所にクロスバスを設けた場合のバス構造の一部を示す図である。図９（ａ）に示すように、ＣＰＵの機能に余力がある場合には、通常使用していない部分にアプリケーションＢの転送機能を持たせるようにしておいて、ＣＰＵによりクロスバスの切り替えを制御する。あるいは、図９（ｂ）に示すように、ＣＰＵに余力がないなどの場合には、ＤＭＡを配置する。このＤＭＡ（ダイレクト・メモリ・アクセス）とは、入出力コントローラと主記憶との間でＣＰＵを介さず直接データをやりとりさせるという転送機能を持ったものである。すなわち、ＣＰＵですべてを行なうのではなく、ＤＭＡという機能を利用し、スイッチング機能を有するクロスバスを設けることで、例

えばアプリケーションＡはＣＰＵで、アプリケーションＢはＤＭＡで処理するような切り替えを行なうことにより、ＣＰＵの負荷率の低減と電流値の分散とを図るのである。この新たに配置するものは転送機能さえ有していればよいのであるが、ＤＭＡの代わりにＣＰＵを配置してもよいことはいうまでもない。そして、同時命令回数がもっとも多い部分に着目し、その型のＣＰＵを採用したときにおける同時命令回数の最高値に応じて、クロスバスを設けないか、一重クロスバスを設けるか、二重クロスバスを設けるかを選択する。

－解析指標－

次に、以上の解析結果で得られる性能のうち最適なものを決定する方法について説明する。本実施形態の解析方法においては、システムの価値を決定する最も重要なパラメータを主パラメータとする。ここでは、性能，電力，面積を主パラメータとする。そして、これらの主パラメータに影響を与える３つの副パラメータを総合し、各主パラメータが目標とする範囲に収まるライブラリ群をまず選び出し、選び出された各ライブラリの主パラメータの値に基づいて、総合的に最適なライブラリ群を選択する方法である。

以下、具体的な方法について順に説明する。

１．交差面積の最小のものを選択する方法

図１０（ａ）～（ｄ）は、各パラメータを満たしているものの中で交差面積が最小のものを選択する方法の手順を示す図である。

まず、図１０（ａ）に示すように、“性能”という共通の主パラメータに影響を与える副パラメータとして、バストランザクションの衝突回数，処理量及び応答時間という３つの副パラメータを設定し、この副パラメータを座標軸とする３次元の座標系を作成する。この３つの副パラメータの値は、各アプリケーションの動作モデルである各ライブラリを用いて動作シミュレーションを行なった解析結果から求められる。例えばＣＰＵに接続される単体アプリケーションＡ，Ｂ，Ｃの構成を、ノイマン型バス構造を採用するか、従来のハーバード系バス構造を採用するか、方向分離型バス構造を採用するかということと、さらに、各アプリケーションの動作におけるハードウェアの割合をどのように設定するかによって、一意的に定まる。例えば、図７，図８に示す解析結果から、バストランザクシ

ョンの衝突回数や処理時間が求まる。このとき、各ＩＯ間のトランザクションの衝突回数の最悪値もしくは平均値、あるいは全体の平均値、ある区間の平均値などをトランザクションＴの点とすることができる。応答時間Ｒは、図７，図８に示すデータ解析を行なったときに実行（シミュレーション）時間である。処理量Ｅは、図７，図８に示す解析を行なったときの実行処理量（合計値）である。

そして、バストランザクションＴが少ないほど、処理量Ｅが少ないほど，応答時間Ｒが短いほどシステムＬＳＩの性能が高いことから、３次元座標系における交差面積が小さいほど性能という主パラメータの１つがベターになる。そこで、この座標系における各座標軸上の各点（副パラメータの値）を結んで形成される平面と座標軸との交差面積がある目標値以下となるライブラリ群を選別しておく。そして、この交差面積を相対的には“性能”の値としても差し支えない。

また、図１０（ｂ）に示すように、“電力”という共通の主パラメータに影響を与える３つの副パラメータとして、処理量Ｅと、ハードウェア化率Ｈと、同時アクティブ密度Ａとを設定し、この３つの副パラメータを座標軸とする３次元の座標系を作成する。ハード化率Ｈは、図６に示すライブラリモデルにおけるハードウェアの割合である。同時アクティブ密度Ａは、図７，図８に示す解析における同時に活性になっているバスの割合である。このときには、処理量Ｅが少ないほど，ハードウェア化率Ｈが小さいほど，同時アクティブ密度Ａが小さいほど電力は少なくて済むので、３次元座標系における交差面積が小さいほど電力という主パラメータがベターになる。そこで、この座標系における各座標軸上の各点（副パラメータの値）を結んで形成される平面と座標軸との交差面積がある目標値以下となるライブラリ群を選別しておく。そして、この交差面積を相対的には“電力”の値としても差し支えない。ただし、この方法には、電力の最大値（ピーク値）で評価するか、電力の平均値で評価するかの２通りの方法があり、いずれでもよいものとする。

さらに、図１０（ｃ）に示すように、“面積（コスト）”という共通の主パラメータに影響を与える３つの副パラメータとして、必要なバス幅Ｂと、必要なメモリ量Ｍと、挿入しなければならないＦＩＦＯ量Ｆとを設定し、この副パラメータを座標軸とする３次元の座標系を作成する。バス幅Ｂは、図６に示すモデルに

おけるバス構造におけるバスの合計幅である。メモリ量Ｍは、図６に示すライブラリモデルで使用するメモリ量である。ＦＩＦＯ量Ｆは、図７，図８に示す解析の結果必要とされるＦＩＦＯ段数の合計である。このとき、バス幅Ｂが小さいほど，メモリ量Ｍが少ないほど，ＦＩＦＯ量Ｆが少ないほど面積という主パラメータがベターになる。そこで、この座標系における各座標軸上の各点（副パラメータの値）を結んで形成される平面と座標軸との交差面積がある目標値以下となるライブラリ群を選別しておく。そして、この交差面積を相対的には“面積”の値としても差し支えない。

そして、図１０（ｄ）に示すように、主パラメータ、つまり、性能，電力，面積を座標軸とする３次元座標系を作成する。そして、図３（ａ）～（ｃ）の処理によって選び出された各ライブラリ群によってそれぞれ定まる主パラメータの値、つまり、性能，電力，面積の値（図１０（ａ）～（ｃ）の交差面積）を示す座標軸上の各点を結んで形成される平面と座標軸との交差面積（３角形の面積）が最小となるライブラリ群を選び出し、このライブラリ群によって決定されるインターフェースを最適なインターフェースとして決定する。

２．特定のパラメータを重視して選択する方法

例えば、図７，図８に示す解析結果から、副パラメータである目標応答時間を満たしているライブラリ群の中で、副パラメータであるバストランザクションが所定値以下のもの、副パラメータである目標応答時間を満たしているライブラリ群の中で副パラメータである処理量が所定値以下のもの、副パラメータである目標メモリ量を満たしているライブラリ群の中で副パラメータであるバス幅が所定値以下のもの、副パラメータである目標ＦＩＦＯ量を満たしているライブラリ群の中で副パラメータであるバス幅が所定値以下のものを選択する。

次に、上記の処理によって選び出されたものの中から、さらに以下の基準によって総合的に最適と思われるものを決定する。この処理には、設計対象の種類等に応じて以下のような方法がある。

第１は、選び出されたライブラリ群の中から、性能，最大電力という主パラメータが目標最低性能，目標最大電力を満たしているものを選別する。そして、その中で主パラメータである面積が最小のライブラリ群を選択する方法である。

第２は、選び出されたライブラリ群の中から、性能，電力という主パラメータが目標最低性能，目標最大電力を満たしているものを選別する。そして、その中で主パラメータである平均電力が最小のライブラリ群を選択する方法である。

第３は、選び出されたライブラリ群の中から、面積，最大電力という主パラメータが目標最大面積，目標最大電力を満たしているものを選別する。そして、その中で主パラメータである性能が最小のライブラリ群を選択する方法である。

第４は、選び出されたライブラリ群の中から、面積，最大電力という主パラメータが目標最大面積，目標最大電力を満たしているものを選別する。そして、その中で主パラメータである平均電力が最小のライブラリを選択する方法である。

３．重み付けを行なって指数化する方法

この方法は、図１１（ａ）～（ｄ）に示すように、主パラメータである性能に対する性能指数ｘ，主パラメータである電力に対する平均電力指数ｙav（又は最大電力指数ｙmx），主パラメータである面積に対する面積指数ｚについてそれぞれ重み付けａ，ｂ，ｃを行なってから加算し、その加算値である最適指数が最小のものを選択する方法である。

ここで、性能指数ｘは、以下の手順により算出する。つまり、図１１（ａ）に示すように、応答時間Ｒ及びその性能影響係数l_xと、バストランザクションＴ及びその性能影響係数m_xと、処理量Ｅ及びその性能影響係数n_xとを算出し、下記式

性能指数ｘ＝Ｒl_x×Ｔm_x×Ｅn_x

により、性能指数ｘを算出する。ただし、応答時間の性能影響係数l_xは、例えば１ｓｅｃのときを“１”とする。つまり、応答時間が３ｓｅｃのときには応答時間の性能影響係数l_xは“３”となる。バストランザクションの性能影響係数m_xは、１０回の衝突があるときを“１”とする。つまり、２０回の衝突があるときには、バストランザクションの性能影響係数は“２”となる。処理量の性能影響係数n_xは、１０MIPSのとき（１MIPSは１００万回の命令数を意味する）を“１”とする。つまり、５０MIPSのときには、処理量の性能影響係数n_xは“５”となる。

電力指数ｙは、以下の手順により算出する。つまり、図１１（ｂ）に示すよう

に、平均処理量Ｅav（又は最大処理量Ｅmx）及びその電力影響係数１ｙと、ハード化率Ｈ及びその電力影響係数ｍｙと、平均同時アクティブ率Ａav（又は最大同時アクティブ率Ａmx）及びその電力影響係数ｎｙとを算出し、下記式

電力指数ｙ＝Ｅav１ｙ×Ｈｍｙ×Ａavｎｙ

ｏｒ＝Ｅmx１ｙ×Ｈｍｙ×Ａmxｎｙ

により、電力指数ｙを算出する。ただし、平均処理量（又は最大処理量）の電力影響係数１ｙは、例えば１０MIPSのときを“１”とする。ハード化率の電力影響係数ｍｙは、２０％のときを“１”とする。つまり、４０％のハード化率のときには、ハード化率の電力影響係数は“２”となる。平均同時アクティブ率（又は最大同時アクティブ率）の電力影響係数ｎｙは、２５％のときを“１”とする。つまり、１０％のときには、平均同時アクティブ率の電力影響係数ｎｙは“０．５”となる。

面積指数ｚは、以下の手順により算出する。つまり、図１１（ｂ）に示すように、メモリ量Ｍ及びその面積影響係数１ｚと、ＦＩＦＯ量Ｆ及びその面積影響係数ｍｚと、バス幅Ｂ及びその面積影響係数ｎｚとを算出し、下記式

面積指数ｚ＝Ｍ１ｚ×Ｆｍｚ×Ｂｎｚ

により、面積指数ｚを算出する。ただし、メモリ量の面積影響係数１ｚは、例えば１ｋByteのときを“１”とする。つまり、１０ｋByteのときにはメモリ量の面積影響係数１ｚは１０である。ＦＩＦＯ量の面積影響係数ｍｚは、１２８Byteのときを“１”とする。つまり、２５６Byteのメモリ量のときには、メモリ量の面積影響係数は“２”となる。バス幅の面積影響係数ｎｚは、１６bit のときを“１”とする。つまり、８bit のときには、バス幅の面積影響係数ｎｚは“０．５”となる。

そして、図１１（ｄ）に示すように、解析指標の判断は、上述の処理によって算出された性能指数ｘ，電力指数ｙ，面積指数ｚについて、それぞれの影響係数（重み付け係数）ａ，ｂ，ｃを決定し、それを加算した値を最適指数とする。つまり、下記式

最適指数Ｏｐ＝ａｘ＋ｂｙ＋ｃｚ

に基づき、最適指数Ｏｐを算出する。そして、最適指数Ｏｐが最小のものを最終

的に選択するのである。

－最適ＩＦ合成－

図１２は、上述の処理を用いて合成される最適ＩＦの構造を示すブロック図である。上述の解析指標に基づいて選択されたバス構造において、ＦＩＦＯを挿入する必要のある箇所にはＦＩＦＯを挿入し、同時処理可能なバス構造を生成するのである。なお、同図において、ＤＭＡを除いて一点鎖線で囲まれた部分がＩＦ（インターフェース）の部分である。すなわち、データバス，制御バス，クロスバス等を含むバス構造と挿入されたＦＩＦＯとにより、ライブラリＡ，Ｂ，Ｃのハードウェアと、ＣＰＵ（動作モデル）と、記憶装置（ＲＡＭ，ＲＯＭなど）とを接続するＩＦが構成されている。

図１３は、本実施形態において説明した最適ＩＦの合成を含むシステム設計の手順を示すフローチャート図である。

まず、データバースとして、図５に示すライブラリＡ，Ｂ，Ｃや、第１の実施形態で説明したハーバード型バス構造，ノイマン型バス構造（デマルチプレクサ型），方向分離型バス構造などを記憶しておく。

そして、ステップＳＴ１で、選択されたライブラリに対してバス構造を選択し、ステップＳＴ２で、システム接続図，転送処理ファイル，新規ライブラリ等を入力する。

次に、ステップＳＴ３で、トランザクションや命令処理の解析を行なう。このとき、図６に示す動作シミュレーションを行なって、図７に示すトランザクション解析と、図８に示す命令処理解析とを行なって、挿入すべきＦＩＦＯの段数や必要なクロスバスの配置などを行なう。

そして、ステップＳＴ４で、例えば上述の３つの手法（図１０，図１１など参照）による解析指標の判断を行なう。このステップＳＴ１～ＳＴ４の処理を繰り返し行なって、最適指数Ｏｐが最小のシステムを選択する。このステップＳＴ３，ＳＴ４においては、ハードウェアとソフトウェアとを併せた性能評価（ＨＷ／ＳＷ性能評価）を行なうのである。

次に、ステップＳＴ５～ＳＴ７で、最適なシステムをハードウェアとソフトウェアとに分割する。すなわち、ステップＳＴ５で、最適ＩＦの合成を行ない、ス

テップＳＴ６で、図１２に示すようなシステム接続図を生成する。一方、ステップＳＴ７で、アプリケーション，フロー制御（ＦＬＯＷ），ＯＳ（オペレーションシステム）などのソフトウェアの選択（条件設定）を行なう。

最後に、ステップＳＴ８で、ハードウェア／ソフトウェアの協調検証を行なう。つまり、ハードウェアを用いてソフトウェアがうまく機能するかどうかを検証するのである。

本実施形態のインターフェースの設計方法によると、アプリケーションについてバス構造ごとに登録された動作モデルとなるライブラリを用いて動作シミュレーションを行ない、この動作シミュレーションの結果から求められる副パラメータ及び主パラメータに基づいて、インターフェースによって接続されるシステム全体の性能を実動作に近い環境で正確に評価することができる。そして、この評価に基づいて、設計者の要望にもっとも適したインターフェースを選択することができ、システム全体の構築に供することができる。

クレーム

1．　半導体集積回路中の制御回路と複数の被制御回路とを接続するためのバス構造であって、

上り側バスと下り側バスとに分離されたアドレスバスと、

上り側バスと下り側バスとに分離されたデータバスと、

を備えていることを特徴とするバス構造。

2．　クレーム1のバス構造において、

上記データバスは、上記被制御回路ごとに分離され、かつ、被制御回路ごとに上り側バスと下り側バスとに分離されている。

3．　半導体集積回路の設計のために利用されるデータベースであって、

制御機能部と複数のアプリケーションとの間を接続するバス構造の種類に関する記述が含まれたテーブルを備えているデータベース。

4．　クレーム3のデータベースにおいて、

上記テーブルは、各アプリケーションの動作モデルによって実現される性能を評価するための性能指数を記述した性能テーブルを備えている。

5．　クレーム4のデータベースにおいて、

上記性能テーブルは、上記性能指数として、処理能力，バス幅，命令量及びメモリ量のうち少なくともいずれか1つのパラメータを含む。

6．　クレーム3のデータベースにおいて、

上記性能テーブルは、バス構造に関する記述として、上り側バスと下り側バスとに分離されたアドレスバスと、上り側バスと下り側バスとに分離されたデータバスとからなる分離型バス構造に関する記述を含んでいる。

7．　複数のアプリケーションの各々に対してその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって、

上記複数のアプリケーションの動作モデルとして上記複数のライブラリを順次用い、

上記各ライブラリについて、上記制御機能部により各アプリケーションを無制

限に動作させる動作シミュレーションを行なって、バストランザクションの衝突回数を解析する。

８．　クレーム７のインターフェースの設計方法において、

各アプリケーション同士の間に上記トランザクションの衝突回数に応じた段数のＦＩＦＯを生成し、ＦＩＦＯを仮挿入した状態で上記解析を行なう。

９．　複数のアプリケーションの各々に対してその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって、

上記複数のアプリケーションの動作モデルとして上記複数のライブラリを順次用い、

上記各ライブラリについて、上記制御機能部により各アプリケーションを無制限に動作させる動作シミュレーションを行なって、同時命令処理の回数を解析する。

１０．　クレーム９のインターフェースの設計方法において、

上記同時命令数に応じてクロスバスの構成を決定する。

１１．　クレーム１０のインターフェースの設計方法において、

さらに、上記同時命令数が所定値以上であるバスに配置するための転送動作の制御機能部を生成し、転送動作の制御機能部を配置した状態で上記解析を行なう。

１２．　複数のアプリケーション及び複数のバス構造の各々に対してその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって、

上記半導体集積回路の価値を最終的に評価するための複数の主パラメータを設定し、上記主パラメータに対して影響を与える複数の副パラメータを設定するステップ（ａ）と、

上記各主パラメータごとに、各ライブラリの副パラメータによって主パラメータを評価して、主パラメータの目標値を満たすライブラリ群を選び出すステップ

（ｂ）と、

上記選び出されたライブラリ群ごとに定まる複数の主パラメータを評価して、最適なライブラリ群を選ぶことにより、インターフェースを決定するステップ（ｃ）と

を含むインターフェースの設計方法。

１３．　クレーム１２のインターフェースの設計方法において、

上記ステップ（ａ）の前に、上記複数のアプリケーションの動作モデルとして上記複数のライブラリを順次用いて、上記各ライブラリについての動作シミュレーションを行なうことにより、上記各ライブラリの副パラメータを解析するステップをさらに含む。

１４．　クレーム１２のインターフェース設計方法において、

上記ステップ（ａ）では、３つの主パラメータを設定し、各主パラメータごとに３つの副パラメータを設定して、

上記ステップ（ｂ）では、３つの副パラメータを座標軸とする３次元座標系を構成して、各副パラメータの値によって定まる３角形の面積が目標値以下のライブラリ群を選び出し、

上記ステップ（ｃ）では、３つの主パラメータを座標軸とする３次元座標系を構成して、選び出されたライブラリ群から求まる主パラメータの値によって定まる３角形の面積が最小のライブラリ群からインターフェースを決定する。

１５．　クレーム１２のインターフェースの設計方法において、

上記ステップ（ａ）の後ステップ（ｂ）の前に、上記複数の副パラメータのうち特定の副パラメータに着目して、特定の副パラメータが目標値を満たしているライブラリ群を選び出し、

上記ステップ（ｂ）では、上記複数の主パラメータのうち特定の主パラメータを除く主パラメータが目標値を満たしているライブラリ群を選び出し、

上記ステップ（ｃ）では、上記特定の主パラメータが最小となるライブラリ群を最適なライブラリ群として選び出す。

１６．　クレーム１２のインターフェース設計方法において、

上記ステップ（ａ）では、主パラメータに対する複数の副パラメータの影響係

数を設定し、

上記ステップ（ｂ）では、影響係数と副パラメータの値とに基づいて各主パラメータの目標値を満たすライブラリ群を選び出し、

上記ステップ（ｂ）では、選び出されたライブラリ群から求まる複数の主パラメータに重み付けを行なってから、ライブラリ群を決定するための演算を行なう。

１７．　複数のアプリケーション及び複数のバス構造の各々に対してその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって、

上記複数のアプリケーションの動作モデルとして上記複数のライブラリを順次選択するステップ（ａ）と、

上記制御機能部により上記各アプリケーションを動作させて、各ライブラリを用いたときの制御機能部，インターフェース及びアプリケーションの性能を解析するステップ（ｂ）と、

上記ステップ（ａ）とステップ（ｂ）とを繰り返し行なって、上記解析結果に基づいて最適なライブラリ群を選ぶことにより、インターフェースを決定するステップ（ｃ）と、

上記決定されたパラメータに基づいて最適なインターフェースを合成するステップ（ｄ）と

を含むインターフェースの設計方法。

１８．　クレーム１７のインターフェースの設計方法において、

上記ステップ（ｂ）では、上記各ライブラリについて、上記制御機能部により各アプリケーションを無制限に動作させたときのバストランザクションの衝突回数を解析し、

上記ステップ（ｄ）では、各アプリケーション同士の間に上記トランザクションの衝突回数に応じた段数のＦＩＦＯを挿入する。

１９．　クレーム１７のインターフェースの設計方法において、

上記ステップ（ｂ）では、上記各ライブラリについて、上記制御機能部により

各アプリケーションを無制限に動作させたときの同時命令処理の回数を解析し、

上記ステップ（ｄ）では、上記同時命令数が所定値以上であるバスにはクロスバスを配置する。

要約

アプリケーションごとに、ノイマン型ＣＰＵ（バス構造），ハーバード型ＣＰＵ（バス構造），方向分離型ＣＰＵ（バス構造）を採用したときの動作を記述した動作モデルとなるライブラリが登録されている。そして、各ライブラリの性能テーブルにおいては、各ライブラリの性能指数が、処理能力，バス幅，命令量，メモリ量というパラメータの関数として表されている。動作のうちソフトの部分とハード化された部分も登録されている。各アプリケーションを順次各ライブラリで置換して動作シミュレーションを行なうことで、半導体集積回路の性能が評価でき、最適なインターフェースを合成することができる。

従来技術

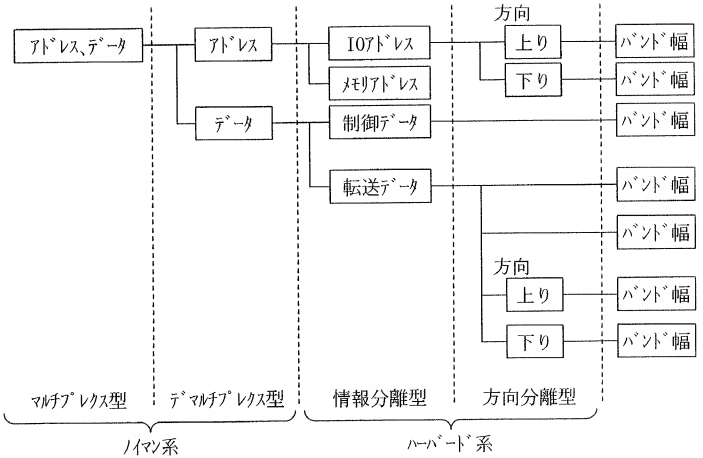
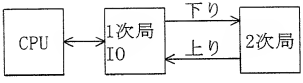
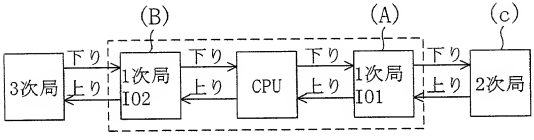


図1

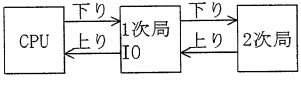
(a)


CPU
1次局
I0
下り
上り
2次局

(b)


(B)
(A)
(c)
3次局
下り
上り
1次局
I02
CPU
1次局
I01
2次局

(c)


CPU
下り
上り
1次局
I0
2次局

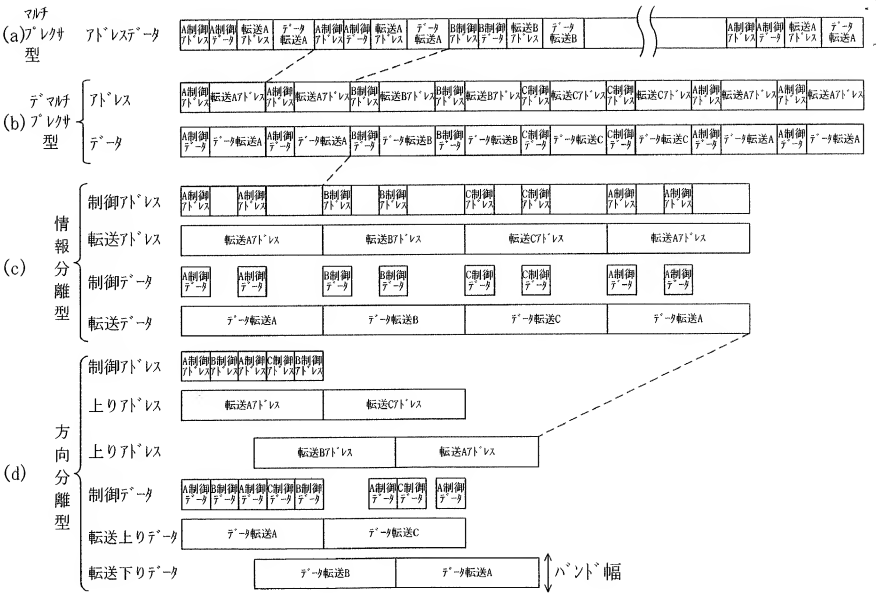
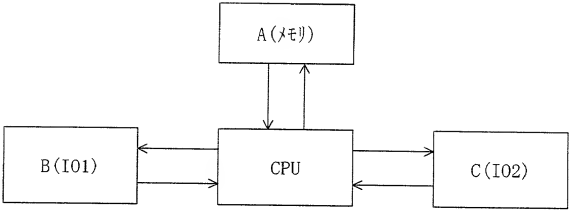


図3

図4



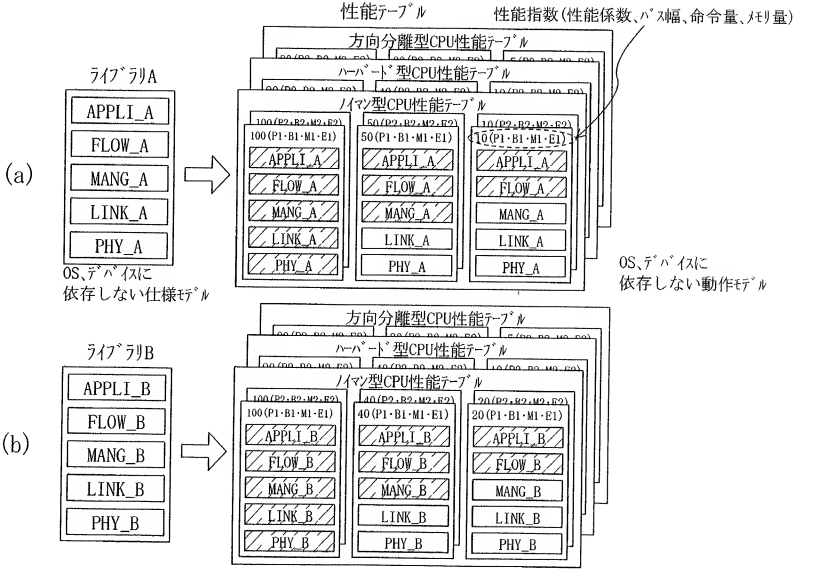


図5

動作ｼﾐｭﾚｰｼｮﾝ

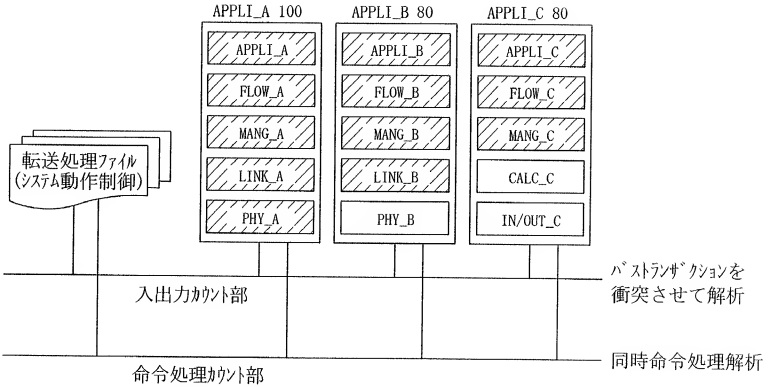


図6

トランザクション解析

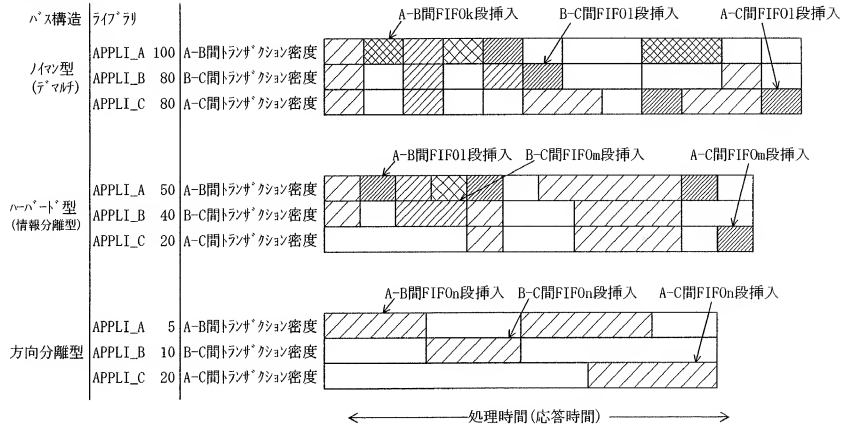


図7

命令処理解析

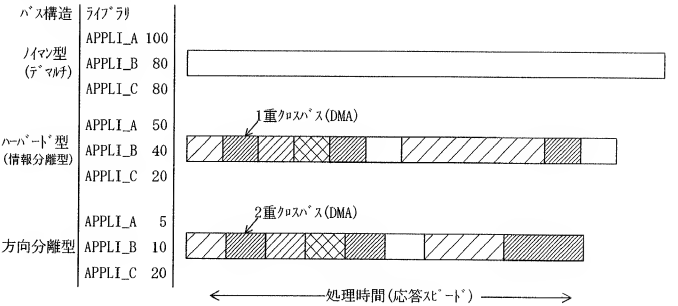
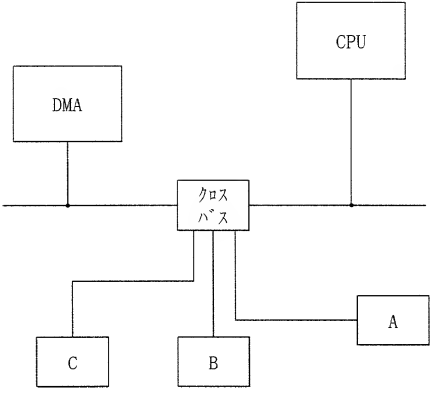


図8

図9



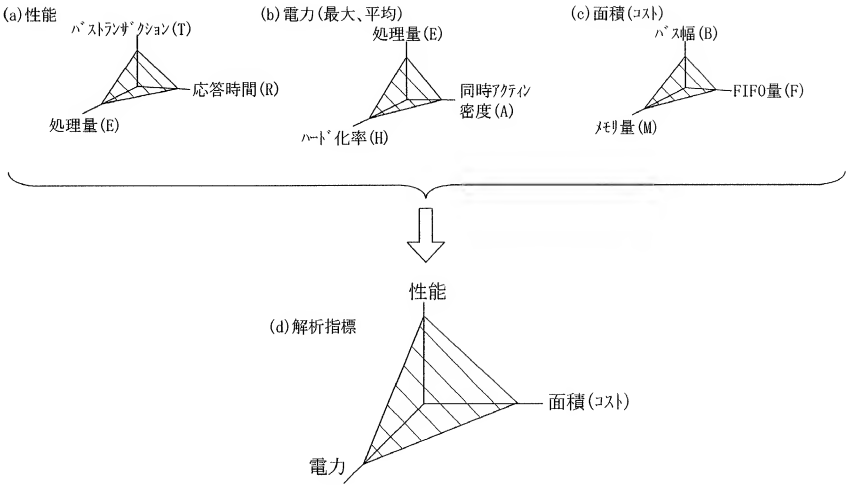


図10

解析指標(重み付け指標)

(a)性能指標の判断基準

応答時間:R　　応答時間の性能影響係数:1x
ﾊﾟｽﾄﾗﾝｻﾞｸｼｮﾝ:T　　ﾊﾟｽﾄﾗﾝｻﾞｸｼｮﾝの性能影響係数:mx
処理量:E　　処理量の性能影響係数:nx
R1x×Tmx×En＝性能指数:x
例)1x＝1/1s, mx＝1/10回, nx＝1/10MIPS

(b)電力指標の判断基準

平均(最大)処理量:Eav(Emx)　　処理量の電力影響係数:1y
ﾊｰﾄﾞ化率:H　　ﾊｰﾄﾞ化率の電力影響係数:my
平均(最大)同時ｱｸﾃｲﾌﾞ率:Anv (Amx)　　同時ｱｸﾃｲﾌﾞ率の電力影響係数:ny
or Eav 1y×Hmy×Aavny＝平均電力指数
Emx 1y×Hmy×Amxny＝最大電力指数 } :y
例)1y＝1/10MIPS, my＝1/20%, ny＝1/25%

(c)面積指標の判断基準

ﾒﾓﾘ量:M　　ﾒﾓﾘ量の面積影響係数:1z
FIFO量:F　　FIFO量の面積影響係数:mz
ﾊﾞｽ幅:B　　ﾊﾞｽ幅の面積影響係数:nz
M1z×Fmz×Bnz＝面積指数:z
例)1z＝1/1kByte, mz＝1/128byte, nz＝1/16bit

(d)解析指標の判断基準

性能指数 (性能)　性能指数の影響係数:a
電力指数 (電力)　電力指数の影響係数:b
面積指数 (面積)　面積指数の影響係数:c
ax＋by＋cz＝最適指数
例)a＝0.5 , b＝0.3 , c＝0.2

図11

最適IF合成

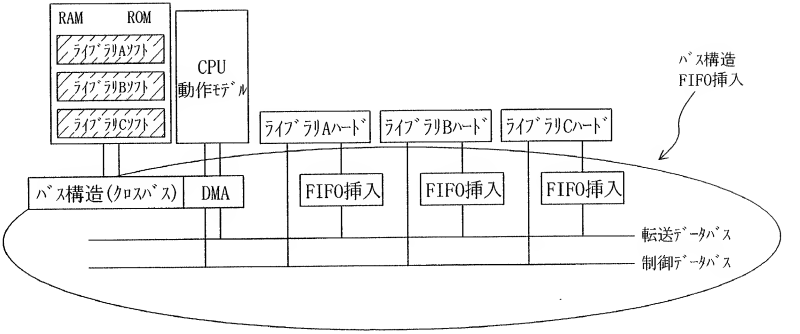


図12

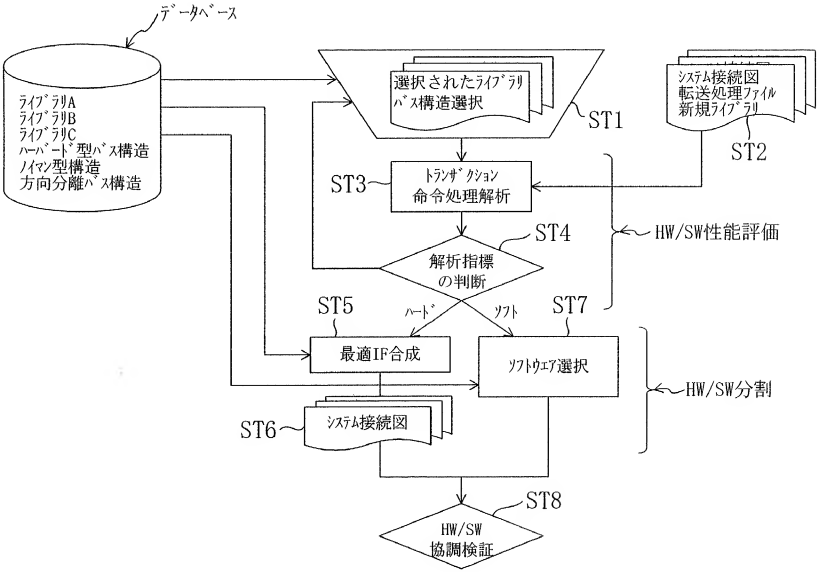

ﾃﾞｰﾀﾍﾞｰｽ
ﾗｲﾌﾞﾗﾘA
ﾗｲﾌﾞﾗﾘB
ﾗｲﾌﾞﾗﾘC
ﾊｰﾊﾞｰﾄﾞ型ﾊﾞｽ構造
ﾉｲﾏﾝ型構造
方向分離ﾊﾞｽ構造
選択されたﾗｲﾌﾞﾗﾘ
ﾊﾞｽ構造選択
ST1
ｼｽﾃﾑ接続図
転送処理ﾌｧｲﾙ
新規ﾗｲﾌﾞﾗﾘ
ST2
ST3
ﾄﾗﾝｻﾞｸｼｮﾝ
命令処理解析
ST4
解析指標
の判断
HW/SW性能評価
ST5
ﾊｰﾄﾞ
ｿﾌﾄ
ST7
最適IF合成
ｿﾌﾄｳｪｱ選択
HW/SW分割
ST6
ｼｽﾃﾑ接続図
ST8
HW/SW
協調検証

図13